AN OVERVIEW OF END–TO–END LANGUAGE UNDERSTANDING AND DIALOG MANAGEMENT FOR PERSONAL DIGITAL ASSISTANTS

R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J.P. Robichaud, A. Celikyilmaz, Y.B. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T. Anastasakos, Z. Feizollahi, N. Ramesh, H. Suzuki, R. Holenstein, E. Krawczyk, V. Radostev

Microsoft Corporation, Redmond, WA 98052

ABSTRACT

Spoken language understanding and dialog management have emerged as key technologies in interacting with personal digital assistants (PDAs). The coverage, complexity, and the scale of PDAs are much larger than previous conversational understanding systems. As such, new problems arise. In this paper, we provide an overview of the language understanding and dialog management capabilities of PDAs, focusing particularly on Cortana, Microsoft's PDA. We explain the system architecture for language understanding and dialog management for our PDA, indicate how it differs with prior state-of-the-art systems, and describe key components. We also report a set of experiments detailing system performance on a variety of scenarios and tasks. We describe how the quality of user experiences are measured end-to-end and also discuss open issues.

Index Terms— personal digital assistants, language understanding, dialog management, Cortana

1. INTRODUCTION

Personal digital assistants (PDAs) and agents with a conversational interface have become a primary focus area for major technology companies [1, 2]. Apple's Siri, Google Now, Microsoft's Cortana and Amazon's Alexa are examples of such systems. Information technology companies are promoting PDAs as gateways to applications and services that provide a meta layer of intelligence that can arbitrate between apps for a given user query. The natural language interface provides high–bandwidth information flow (compared to touch and typing) between the user and the agent, and can lead to faster task completion, ultimately improving the user's productivity. Typically, a PDA provides both proactive and reactive assistance. With proactive assistance, the system takes an action based on the events it has been tracking. Events can be based on time, location, physical or digital activity, or any combination thereof, *e.g.* informing a user they should leave soon to catch a flight. With reactive assistance, the system responds to the user's explicit spoken or typed request. In the remainder of this paper we use the term PDA(s) to mean the reactive assistance functionality unless explicitly stated otherwise.

Over the past three decades, conversational understanding has been studied both in academia and corporate research laboratories [3, 4]. Today, most large corporations use a conversational understanding system in their call centers to handle customer requests. Such systems are generally not open ended; each is designed to handle a limited set of scenarios, typically scoped to the business of the company. Similarly, in an academic setting, many systems have been developed that handle complex dialog interactions with users but are typically limited to a small numbers of domains [5, 6].

PDAs push the boundaries of natural language understanding (LU) and dialog management, moving beyond both traditional research systems in terms of the breadth of domain coverage, and beyond corporate systems by having to support multiple inhomogeneous services at scale. The principal scientific challenge in building such a PDA is that the user is unconstrained in what they may say. Essentially, the problem is one of how to construct an open domain language understanding and dialog system. This paper presents an architecture developed with the dual aims of realizing a practical system, while moving towards an answer to the challenge of open domain dialog by combining multiple back-end services. We describe how these design considerations have impacted both the overall architectural and the individual platform components.

The most similar system to the one described here that has been published is that presented in [7]. The principal difference is that in the system they describe, the language understanding and dialog for each domain are handled by independent domain expert systems. In the system we describe here, language understanding and some of the dialog components are centralized to provide additional cross-domain flexibility and re-use, improving the system's ability to ramp up support for new domains.

The rest of the paper is organized as follows. We first introduce a set of design requirements and the complex interactions between them, in section 2. Section 3 describes the system architecture. Details of the system are presented in section 4 (language understanding) and 5 (dialog management), respectively. Section 6 presents the experimental results. The last section concludes with a short discussion and future work.

2. PDA REACTIVE ASSISTANCE REQUIREMENTS

From a user perspective, the high level requirement is that a PDA should understand everything and be able to do almost anything, whereas developers aim to match or exceed user expectations but at an acceptable cost. For an industrial scale PDA platform, this results in a multi-dimensional optimization problem that includes:

1. the breadth of LU domains and experiences;
2. the naturalness of user language;
3. the complexity of dialogs;
4. the range of modalities and devices the PDA can interact with;
5. the supported range of expertise of experience authors;
6. the latency and capacity of back-end or cloud services that can be accommodated;

All opinions are those of the authors and should not construed as representing the official position or plan of action of Microsoft. The authors would like to thank Vipul Agarwal, Danko Panic and Ravi Bikkula for many helpful discussions.

7. the overall latency and accuracy of system responses;
8. allowable costs, *e.g.* computational, implementation, and maintenance;
9. support for development of uniform user experiences, *e.g.* the PDA "personality";
10. support for easy upgrading of experiences.

We discuss the interactions between the various dimensions below.

In this paper we define an *experience* as the PDA equivalent of a smart phone or desktop application, *i.e.* the smallest unit of self–contained interaction that fulfills a user's need and is implemented by a single developer or single team. Typically, a user's need is fulfilled by presenting information or completing a transaction; however, for some experiences success is defined as continued engagement (*e.g.* for games or chat bots). Larger "experiences" can be built up through interleaving and chaining of smaller experiences.

We define an *LU domain* as a collection of related intents and slots for which no operational conflict arises in the semantic space that they span. A domain can cover many related intents, *e.g.* a "local" domain can include searching for local attractions, getting directions, checking traffic conditions, and making restaurant reservations. Other domains are defined more narrowly to support a single task, *e.g.* pizza ordering. Each experience is typically served by a single domain, while one domain may support multiple experiences.

A claimed selling point of PDAs is that they can enable users to get many things done via a single entry point – no more searching for a forgotten app [1]. This justifies maximizing dimension 1, that a PDA should offer wide coverage in terms of both what it can understand and what experiences it enables.

The naturalness of the user's language that can be understood (dimension 2) can vary widely. This is especially true for the first turn in an interaction, where the system has to decide on the user's intent and thus which experience they wish to engage with. Options range from imposing strict constraints on the supported language, requiring users to explicitly refer to the application name before issuing a keyword query, *e.g.* "OneBusAway! Next bus home.", through to parsing natural language queries such as: "When's the next bus from work to home".

The complexity of the supported dialogs (dimension 3) is itself a multi-dimensional space:

- *conversation structure* ranges from unstructured experiences (*e.g.* single-turn responses that present query results), multi-turn browsing/search refinement, chit-chat style dialogs, through to goal-orientated, transactional interactions;
- *allowed forms of initiative*; beyond who can influence the dialog flow, *e.g.* system, user or mix initiative dialogs, limits may exist on the form of initiative that can be taken. A browsing scenario can be user initiative only, but even then the user's initiative can still be constrained to specifying keywords/filters, not supporting certain interrogative queries, e.g. "what's the phone number?". The system can be completely passive, only presenting results, can be mostly passive with the exception of disambiguation turns, "I found three locations. Which do you want?", or actively lead the dialog when it believes information should be collected, *e.g.* "How many tickets do you want?" The latter case can exhibit purely system driven flow, or can allow the user varying degrees of initiative, *e.g.* provision of out-of-turn information, task switching, *etc.* The forms of initiative allowed directly impacts dimension 2 and modifies the risk of errors (dim. 7);
- *information sharing between experiences*; options include treating each dialog as being independent with no sharing of information, long-term storage of information related to individuals, short term storage and passing of information between different experiences. The last option allows for interleaving or chaining of dialog experiences, *e.g.* carrying over flight destination into a hotel booking task;
- *automation level* ranges from fully-automated dialogs to human-in-the-loop, the latter allowing more complex queries to be handled by a human agent. This directly impacts the trade-off between latency and accuracy – dimension 7.

The range of modalities (dimension 4) and by extension the range of devices supported also tie in with dialog complexity. Each device's capabilities and form factor will allow for different modalities; each experience should thus operate seamlessly on and use a dialog style that makes best use of those modalities.

No single person can author sufficient experiences to achieve the breadth to maximize dimension 1. Thus, a distributed approach to authoring experiences is necessary, leading to consideration of dimension 5. Experience authors may include conversational understanding experts, non-expert in-house collaborators, external third-party developers, and users themselves teaching their own PDA. Enabling non-experts may require different trade-offs, particularly related to the sophistication of the authoring tools.

Back-end databases and services (in-house or in the cloud) are essential in delivering meaningful experiences, *e.g.* checking upcoming meetings or booking a cab. These services vary widely in their capabilities, especially in terms of throughput and latency. This leads to consideration of dimension 6, in that the platform must accommodate the range of capabilities of each service used. This impacts system design, in particular with regards to latency and accuracy (dimension 7) and costs (8). For instance, slow back-end services may be used only infrequently, or only when the users' intent is clear, even though having the results from these services (or knowing that no results exist) ahead of time would help identify the user intent.

The trade-off between latency, accuracy and costs (dimensions 7 and 8) manifests itself beyond the interaction with back-end services, at all levels of the system. One such trade-off involves culling alternative interpretations early to reduce computational cost, versus carrying multiple hypotheses throughout system *and* process them in parallel to boost accuracy while not impacting latency. Implementation and maintenance costs can also have a bearing on system architecture. While machine learnt models may yield better accuracy, a developer might prefer to use a hand coded set of rules whose operation is easy to understand by untrained staff, despite the potential loss in accuracy or generalization. The underlying computations should thus be appropriate given the level of sophistication of the experience, to minimize complexity of implementation and debugging.

Uniformity of experience tends to improve usability and user engagement. Given that maximizing dimension 1 tends to lead to a distributed approach to authoring, then support for developing uniform user experiences (dimension 9) becomes essential, in particular when opening up development to third parties. Support for uniformity of experience can be achieved by providing predefined visual templates that can serve a range of tasks, as well as tools to automatically generate written and spoken prompts with the appropriate agent "personality" given the context of the interaction.

The upgradeability dimension 10 refers to both the ability to refresh experiences (*e.g.* ease/speed of joint optimization and robustness to rapid changes in users' queries, such as recognition of trending terminology) as well as an upgrade path for increasing the level of sophistication of experiences (*e.g.* minimizing the effort to turn a single turn user-initiative dialog into a multi-turn mixed-initiative one). This is important in terms of the longer term developer experience and developers' willingness to invest time and effort; the ability to bootstrap a new experience quickly and then improve it over time is viewed as essential by many third-party developers.

Given this broad set of dimensions and the interplay between them, it is clear that many design choices must be made. The problem is further complicated by different demands being placed on different experiences. In the end, a PDA platform has to offer a mix of solutions that land at different points in this multi-dimensional space and yet make them function together as a larger whole.

3. SYSTEM ARCHITECTURE

In line with maximizing dimension 1, which is seen as essential for commercial PDAs, our system covers a broad range of experiences, including: web search; chit-chat; question answering; commanding on-device functionality such as setting up alarms and timers, playing music, sending SMS, and document search; as well as interacting with other cloud services such as checking movie hours and getting directions to or checking the weather at a particular location.

An overview of our PDA architecture is shown in Fig. 1. The user sends a request to the PDA using spoken or typed input. The system interprets the request and generates a response. The system functionality is grouped into three broad categories:

1. input processing, including LU;
2. updating the dialog state, and
3. applying a policy to select and execute the system action,

with multiple hypotheses tracked through each of the stages.

This diagram somewhat simplifies the actual architecture as in reality a plurality of parallel data flow paths exist. The diagram does however accurately capture that all these flows are one way and arrive at a ranking and selection stage. The diagram also illustrates one of our design decisions, that to support a range of expertise in experience authoring (dimension 5), plus making upgrading experiences to multi-turn easy (dimension 10), while simultaneously managing runtime costs (dimension 8), the platform is broken into distributed, independent modules. New experiences are strongly encouraged to reuse the signals provided by upstream modules. A tension that arises with this design is where experience–specific requirements should be encoded. This tension is felt throughout the platform as developers' expectations are that upstream modules should behave the way their experience expects it. Such tensions can, for example, lead to splitting an LU domain to better serve multiple experiences without conflicts. Our design contrasts with the approach described in [7], where such tension is relieved by having multiple self contained conversational systems, each with their own LU. The core platform in that case is responsible only for routing of user queries. Our design promotes ease of authoring and sharing of modules at the cost of a more complex architecture.

In terms of experiences, the platform adopts a layered approach. Similar to other PDAs [7], the base layer is a web search service which serves as the fall-back experience for queries which cannot be processed by a more specific provider. The next layer are question-answering services, covering a variety of domains, that directly answer user queries without requiring them to navigate to related websites. To provide a more sophisticated multi-turn experience (dimension 3), this layer can benefit from carrying over contextual information from previous turns, especially to answer contextual questions, such as "How tall is he?". The slot/entity carry over (SCO) module, section 5.1, provides such functionality. The question answering layer, while not as broad as general web search, provides significantly broad topical coverage. Other layers use flexible item selection (FIS) (section 5.2) to take system initiative disambiguation turns, *e.g.* prompting the user to select between a number of items; "Here are some nearby Mexican restaurants, which do you want directions to?", also see figure 2. Other layers make use of session storage capabilities (not shown). Among such experiences are game-playing and chit-chat based systems, as well as transactional dialogs. The latter are system- or mixed-initiative experiences which lead the user to the execution of an intended task.

The output of the service providers are ranked [10, 11, 13], and a dialog policy component determines the final system response, which can include a visual card containing the content/entity information or an action. Typically, for voice input, the system also generates a natural language response, which can be synthesized into speech with a text–to–speech (TTS) synthesis engine. Key components of the system are described in detail in the next two sections.

4. LANGUAGE UNDERSTANDING

The language understanding (LU) component uses as input either typed text or speech transcription from an open vocabulary speech recognition service and performs semantic analysis on the query to determine the underlying user's intent [8, 9, 4, 15].

Our PDA LU is built to handle multi-domain, multi-turn, contextual query understanding [14, 9, 15, 16], subject to the constraints of the back-end data sources and experiences. The LU module semantically parses and analyses the queries according to a domain-specific semantic schema. Given a requirement to minimize latency and computational costs (dimensions 7,8) deep semantic analysis is not performed. Instead, LU delivers a flat analysis matching the experience requirements while allowing for free form natural language expressions that represent different user intents and slots.

4.1. Domain, Intent, Slot Modeling

Prior work has shown a variety of approaches can be used for language understanding [17, 8, 20, 21, 15]. Some early systems used rule based parsing, hand-authored by an expert for a specific domain [17]. Recent state-of-the-art systems use machine learned models [21, 20, 15]. Semantic meaning of a query is represented as a semantic frame (SF), which is a tuple of $\langle$DOMAIN, INTENT, SLOTS$\rangle$ in the semantic space. The slots are a list of key-value pairs: $\langle$SLOTNAME, SLOTVALUE$\rangle$. In our system, domain classification is done using machine learned (ML) classifiers such as SVMs [25], which generate a score for each domain. We use a separate binary one-against all classifier for each domain. For each domain, we also train a domain specific intent and slot model. Intents and slots can be shared across different domains. Intent detection also uses SVM classification but is framed as a multi-class classification problem. Slot tagging is considered as a sequence classification problem. Conditional random fields (CRFs) [26] and more recently deep learning techniques [9, 31, 16] are used for slot tagging. All of the LU models use weighted entity lexicons [18, 19] and are additionally trained with artificial out-of-vocabulary (OOV) words to improve generalization to unseen entities. The slot values can be further resolved into entities (*e.g.* a strongly typed object in some back-end data source) or canonicalized into a standard form (*e.g.* time/date values). To deal with the open domain nature of queries, each LU domain uses training data from all other domains as negative examples, including representative data for the web domain

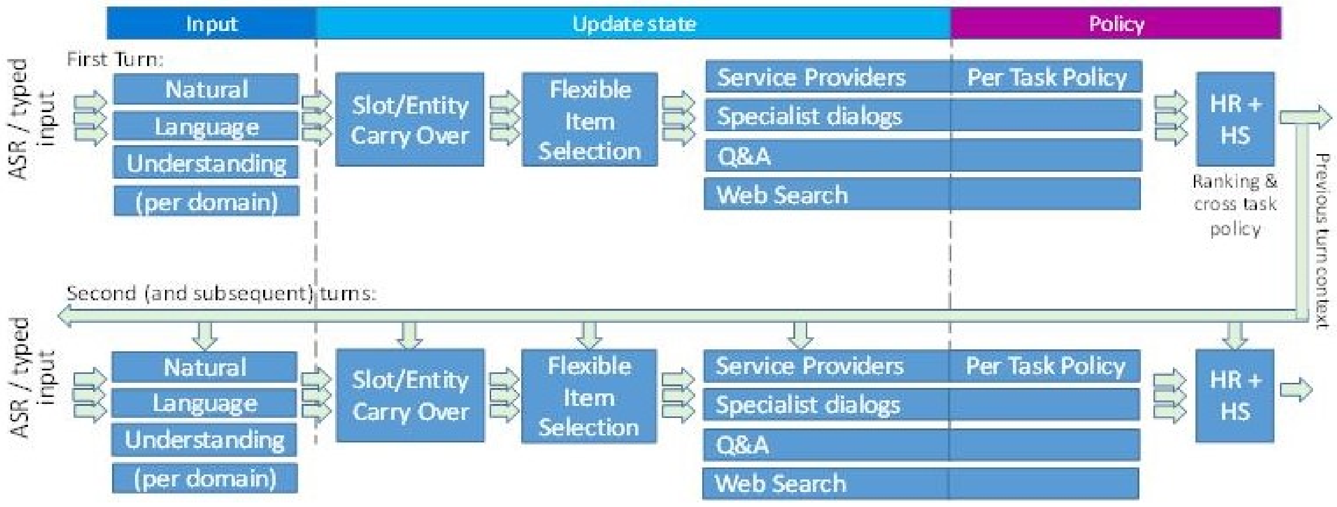


Fig. 1. Personal Digital Assistant Reactive System Architecture.

which is treated as representing everything else in the world that the PDA does not explicitly have a response for. At run-time, LU does not filter the results. Multiple alternate LU semantic analysis (at least one for each domain) are generated and processed throughout the rest of the pipeline allowing late binding on the user's intent [10].

Follow-up queries in isolation are ambiguous, *e.g.*,

- Turn 1: " how is the weather in New York" (weather)
- Turn 2: "what about the weekend" (weather)

or

- Turn 1: "how is my schedule" (calendar)
- Turn 2: "what about the weekend" (calendar)

LU modeling thus must be performed in a contextual manner. Here, conversational session information from the history greatly reduces the ambiguity of the current turn. We built a set of models addressing contextual domain, intent and slot modeling [14, 15, 16]. Our current approach involves modeling contextual intent prediction as a sequential tagging problem with previous utterances as features that condition this turn's prediction. Contextual information from beyond LU, such as dialog state (where this is tracked by an experience), and PDA responses are also exploited as effective external features. Contextual modeling of queries reduces the likelihood of abrupt intent or domains switching leading to more coherent interaction during a multi-turn session.

5. DIALOG

The collection of LU models' analyses are passed to multiple experience providers, as seen in figure 1. Experience providers are called in parallel to processes the query, make calls to specific backend knowledge sources and build appropriate responses. SCO and FIS are examples of built-in platform components that are available to experience providers to allow easy reuse of common functionality. They enable service providers to provide multi-turn support, and thus aid in platform extensibility.

5.1. Slot/Entity Carry Over (SCO) and Co-reference

Tracking keywords contained in slots and associated entities enables a certain set of user initiative experiences, *e.g.*

- Turn 1: "find french restaurants in seattle"
- State 1: cuisine="french", place_type="restaurants", absolute_location="seattle"
- Turn 2: "how about chinese"
- State 2: cuisine=**"chinese"**, place_type="restaurants", absolute_location="seattle"

SCO decides which slots from previous turns are still relevant in the current turn of a multi-turn conversation. It also models implicit start-overs, *i.e.* dropping of all slots, and operates both within and across domains powering interactions such as "What's the weather like at Snow Lake?", "Get me driving directions."

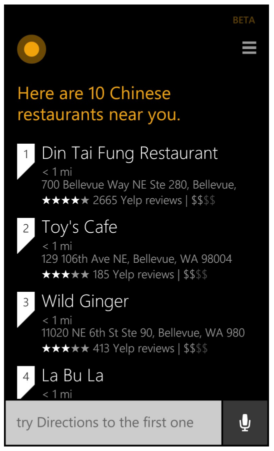
We use a hybrid rule-model approach to implement our SCO system, with the set of rules serving as baseline with a generalized ML model to scale better to new scenarios. Model-based SCO is implemented as a supervised learning task [22]. For each slot from previous turns, the model decides to carry-over or drop slots. Slots in the current turn are unconditionally kept. Our implementation of SCO is as a gradient boosted decision tree, where the concatenation of the current and previous turn utterances are compared to indicate relevant slots. At least one most likely tagging is passed on for further processing. If multiple taggings are produced by SCO, the hypothesis ranking and selection (HRS) component is the ultimate determiner of the winning tagging.

For experiences that use strongly typed entities, *e.g.* celebrity question-answer experiences, an alternative co-reference resolution model exists that learns relationships such as 'him' or 'her' from mining knowledge bases, *e.g.* Satori – Microsoft's equivalent of Freebase or Google's Knowledge Graph.

5.2. Flexible Item Selection

Unlike traditional over-the-phone spoken dialog systems, PDAs run on different device and form factors (*e.g.* smartphones, PCs and browsers). They have the visual element as an additional modality to communicate the system's response to the user. Visual display of the system's response changes human behavior when interacting with PDAs. In Fig. 2, we show the PDA response to the query "*show me chinese restaurants nearby*". The user can either select one (or more) of the items or choose none and instead reformulate their query. Thus immediately following a PDA's attempt to elicit a selection from a user, the system has two decisions to make; whether the user's utterance is intended to be a selection, and what the user is selecting. The former problem is handled later in HRS as it can be cast as part of the larger problem of determining if a user has switched tasks. The FIS model solves the latter problem.

There are a number of ways a user can refer to the items on the screen; item number, full or partial title, or the metadata shown for each item (*e.g.* address, rating). We pose the problem as a classification task to correctly identify intended on-screen item(s) from user utterances [23]. The FIS model is a gradient boosted decision tree (GBDT) trained as a binary classifier. It jointly scores the pair of the user's utterance (referring expression) and information from the candidate item, for each item on the list presented to the user. Items are scored individually and a threshold is applied to indicate those selected. FIS is trained with human annotated data.


BETA

Here are 10 Chinese restaurants near you.

1 Din Tai Fung Restaurant
< 1 mi
700 Bellevue Way NE Ste 280, Bellevue,
★★★★★ 2665 Yelp reviews | $$$$

2 Toy's Cafe
< 1 mi
129 106th Ave NE, Bellevue, WA 98004
★★★★★ 185 Yelp reviews | $$$$

3 Wild Ginger
< 1 mi
11020 NE 6th St Ste 90, Bellevue, WA 980
★★★★★ 413 Yelp reviews | $$$$

4 La Bu La
< 1 mi

try Directions to the first one

Fig. 2. Flexible on-screen item selection

5.3. Experience Providers

Experience providers are a set of non-homogeneous services that share a common input and response API. Each can implement its own dialog policy under the hood and optionally use the platform provided SCO and FIS to power their scenarios. As described at the start of Section 3, the service providers are layered in such a way as to provide both deep interactive experiences and the necessary shallow but broad coverage to handle the open domain nature of first turn queries. The base layer with the broadest coverage is web search, followed by a multitude of question-answer services that respond with single turn and contextual responses (context from previous turns). More specialist providers will engage users with system initiative turns such as selection across multiple items (*e.g. system: "Here are 10 Chinese Restaurants near you, which one do you want?"*) making use of FIS; even more sophisticated scenarios, *e.g.* 20 question games, chit chat or task orientated dialogs will use complex, custom modes of interaction [12]. Interfacing to back-end data sources is also done through experience providers.

This multitude of experience providers execute in parallel, taking advantage of the range of signals from the multiple upstream LU models, SCO and FIS. What is ultimately shown to the user depends on the ranking and selection component, HRS, discussed next.

5.4. Hypothesis Ranking and Selection

HRS is the mechanism by which the platform ranks and selects the final response shown to the user. The processing is done in two stages, with an initial ranking of the hypotheses (HR) followed by selection of the best hypothesis according to rank score and other features (HS). One of the key decisions in dealing with open domain requests is the recognition of requests that are not handled by any specialized experience provider and letting such requests fall through to web search.

Ranking is carried out over dialog hypotheses, where each dialog hypothesis is an assembly of the SF and an experience provider response (*e.g.* back-end knowledge plus suggested dialog act). Making the selection of the system response as late as possible has the benefit that the ML model has full access to all LU analyses, as well as the full session context and knowledge of whether a particular experience produced a response. An experience is allowed to not respond. HRS is contextual in that it should learn to favor the continuation of an existing multi-turn dialog unless the user's utterance indicates a strong likelihood of wishing to switch domains.

The HR models are trained using GBDT. For training, each dialog hypothesis is assigned a rating of 1, if its domain matches that selected by an annotator, and 0 otherwise. The HR model score is then optimized using LambdaRank [28] to maximize the likelihood of it ranking a hypothesis with rating 1 in the top position. Over 1,000 features are extracted for each hypothesis. These include: binary features that indicate the presence or absence of a particular entity tag in that domain's analysis of the user's utterance, the domain's interpretation of the intent, *etc.* We also use contextual features such as whether the hypothesis's domain matches the top ranked domain from the previous turn, how many entity tags a hypothesis has in common with the previous top ranked hypothesis, as well as the complete list of previous turn domains scores. Features extracted from back-end domain knowledge include whether or not results can be generated for that experience from the SF corresponding to that hypothesis. Input features to HR models are mostly derived features in the semantic space, *e.g.* the existence of a slot tag but not the actual words tagged, and are thus not language dependent. Therefore, the HR model can be trained for use across multiple languages and locales [11].

HS encodes meta-scenario policy decisions. In general, the top ranked item by HR is selected. However, sometimes it is desirable to override the ranking, *e.g.* if the two highest rank hypotheses are very close then it might be better to confirm the user's intention rather than just picking the top one.

6. EXPERIMENTS

6.1. Data

To measure system performance, we use an internal dataset collected from interactions between real users and the Cortana personal digital assistant. The data was sampled randomly from Cortana system logs at regular spaced intervals over a period of several months such that the final distribution was 80:20 speech vs. text queries. We use English (US) data that spans 9 distinct domains, with a distinct provider responsible for serving each corresponding experience. A catch-all "Web" domain includes all data not covered by the other categories, including targeted question-answering, chit-chat, and general web queries. Details about each domain are shown in table 1.

6.2. First-Turn Performance of LU and HRS Systems

As a first experiment, we examine the performance of the initial LU analysis as well as that of the end-to-end system. We split the available data 70-20-10 into LU training, HRS training and final result evaluation. The LU models were trained on the LU training set only. To minimize over-reliance on LU annotations during ranking, HRS was trained using both the LU training and the HRS training sets. The HRS training process used automatic LU labels plus experience responses generated from the pipeline, resulting in a set of training examples with input features required by the HRS model and with human annotated domain labels as the supervisory signal.

Experimental results are shown in table 2. For each domain of interest (other than the catch-all "Web"), we present domain and in-

Table 1. *Data discussion*

Domain	Supported Experiences
Alarm	setting, querying, and deleting alarms
Calendar	creating and querying calendar entries
Communication	making phone calls, sending emails and SMS
Device Settings	commands for changing local device settings
Document	document search on the local filesystem
Entertainment	music playback
Local	traffic information, driving directions, search for businesses and attractions
Reminder	setting, querying, and deleting reminders
Weather	weather information
Web	general web search, chit-chat, and targeted question-answering not covered above

Table 2. *Domain Accuracy, Intent, Slot F1 Measure, Semantic Frame (SF) Accuracy for LU / HRS.*

Domain	Dataset Size	Domain Acc.	Intent Acc	Slot (F1)	Semantic Frame
alarm	12938	97.5 / 97.6	94.4 / 94.5	95.8 / 95.8	91.0 / 91.1
calendar	12296	90.1 / 90.6	86.9 / 87.3	88.3 / 88.2	79.6 / 79.9
communication	58837	91.2 / 95.2	86.0 / 89.6	80.0 / 80.6	74.8 / 76.7
entertainment	8369	92.9 / 92.4	92.4 / 96.2	90.0 / 89.8	79.6 / 79.2
device settings	21102	35.7 / 84.8	35.9 / 39.6	40.6 / 81.6	32.3 / 36.7
document	4491	76.9 / 86.3	74.4 / 83.5	83.5 / 86.8	61.1 / 67.6
local	45075	92.2 / 95.0	90.0 / 92.7	87.3 / 88.4	77.1 / 79.1
reminder	26612	90.5 / 92.5	89.6 / 91.5	89.6 / 89.9	82.8 / 84.4
weather	26613	98.5 / 98.6	96.9 / 96.9	95.0 / 95.0	90.4 / 90.4
web	227939	96.0 / 97.1	N/A	N/A	96.0 / 97.1
Overall	444272	86.2 / 93.1	82.9 / 85.8	84.6 / 89.3	76.5 / 78.2

tent classification accuracy, slot tagging F-measure, as well as the aggregate SF accuracy. For the "Web" domain we present only domain accuracy, since the chit-chat, question-answering, and general web search experience providers are not powered using an ontology of intents or slot tags. We find that in most cases HRS improves upon the LU results. Largest differences can be observed in the "Documents" domain, which is significantly less frequent in our data, and the "Device Settings" domain. Analysis of the confusion matrix of the LU results shows that "Device Settings" is most confusable with the "Web" domain; taking into account the features extracted from the overall conversation state helps HRS minimize the confusability between these two domains.

6.3. End-to-End System Accuracy

While HRS performance is a good indicator of the ability of the system to understand and make good decisions about which experience provider's output to present to the user, it does not measure the overall system performance. In particular, decisions made within each provider's component are not captured by the HRS metric. Instead, end-to-end (E2E) task success studies have been shown to be a good indicator of user (dis)satisfaction with the system [29, 30].

We perform an additional study to measure E2E task success using human judgements. For this experiment, a subset of the test queries are selected, and the final system output is captured. Human judges are shown both the queries and the final system responses. Each query-response pair is assigned a score ranging from 1 (terrible) to 5 (perfect). The scores are converted to a binary decision by mapping ratings of 3 and above to a **success** state. E2E success metric is thus computed as the proportion of successful queries.

Table 3. *End-to-end system accuracy using human judgements.*

Domain	E2E Success (Full)	E2E Success (Filtered)
alarm	64.3	92.9
calendar	92.9	92.9
communication	68.5	72.4
device settings	78.6	78.6
local	88.6	88.7
reminder	68.7	87.5
weather	96.5	96.5
web	85.4	85.4
Overall	80.4	86.9

Results for this experiment are shown in Table 3. Two configurations are shown. **Full** includes the complete set of queries used for the E2E analysis. **Filtered** includes only queries whose ground-truth (human-labeled) intent can be supported by the existing system providers. Comparing the two sets of numbers allows us to determine whether user dissatisfaction is related to system mistakes or missing system capabilities. One such example can be seen in the case of the **alarm** domain. The **Full** configuration shows a relatively low E2E success rate. However, much of the dissatisfaction is related to the system inability to service user queries such as "delete my 6am alarm". In this case, even though the LU component correctly tags the utterance with the *delete_alarm* intent, the alarm provider at the time of testing was incapable of handling this request and the user is shown a set of web results instead, leading to user dissatisfaction. When utterances whose ground-truth intent is unsupported are filtered out, the E2E success rate for the **alarm** domain improves, showing that the system does a relatively good job of processing the queries which it is capable of handling.

7. CHALLENGES AND DISCUSSIONS

In this paper, we have attempted to summarize the many dimensions and constraints that are faced when developing a large scale PDA architecture, and we have proposed a PDA architecture that attempts to address these with an eye towards expansion and component reuse. We demonstrate experimentally that performing late binding allows the system to correct mistakes made during the initial LU analysis through use of additional features extracted from the dialog state and experience-specific providers. The subjective scores obtained from end-to-end analysis show that much of the user dissatisfaction is caused by the system's inability to service queries which had been understood correctly, rather than mistakes during the understanding or ranking components - in other words, due to engineering or other practical limitations, rather than fundamental architecture decisions.

Many unsolved challenges related to language understanding and dialog management for PDAs still remain. On the LU side, a key issue is that of domain scaling, *i.e.* the ability to understand everything a user might say. In particular, quick ramp-up of LU models for a new domain, starting with minimal or no in-domain data, is of great importance to PDA developers. On the dialog management side, of great importance is the ability to develop domain-independent state tracking and policy models, which can then be reused across all new experiences. From an engineering perspective, heterogeneous back-ends and application interfaces remain bottlenecks for expanding to new domains, as each new back-end requires custom query building and processing of results. Addressing these issues will likely require new trade-offs on the continuum of dimensions that span the requirements for a successful PDA.

8. REFERENCES

[1] R. Sarikaya, "The technology powering personal digital assistants", *Interspeech Keynote Presentation: http://www.superlectures.com/interspeech2015/the-technology-powering-personal-digital-assistants*, Dresden, Germany, 2015.

[2] SRI International. CALO: Cognitive Assistant that Learns and Organizes. pal.sri.com, 2003-2009.

[3] D. Goddeau., E. Brill, J.R. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff and V. W. Zue, "Galaxy: A human-language interface to on-line travel information", *In Third International Conference on Spoken Language Processing*, 1994.

[4] N. Gupta, G. Tür, D. Hakkani-Tür, S. Bangalore, G. Riccardi and M. Gilbert, "The AT&T Spoken Language Understanding System", IEEE Trans. Audio, Speech and Language Process. v. 14, no: 1, January, 2006.

[5] D. R. Traum and S. Larsson, "The Information State Approach to Dialogue Management", In J. van Kuppevelt and R. W. Smith, editors, *Current and New Directions in Discourse and Dialogue*, pp. 325-353, Springer, 2003

[6] S. Young, M. Gašic, B. Thomson and J. D. Williams, "POMDP-Based Statistical Spoken Dialog Systems: A Review", *Proc. of the IEEE*, vol. 101, no. 5, pp. 1160-1179, 2013.

[7] Z. Wang, H. Chen, G. Wang, H. Tian, H. Wu and H. Wang, "Policy Learning for Domain Selection in an Extensible Multi-domain Spoken Dialogue System", *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing EMNLP*, 2014.

[8] G. Tür and R. De Mori, Eds., "Spoken Language Understanding: Systems for Extracting Semantic Information from Speech", New York, NY: John Wiley and Sons, 2011.

[9] A. Deoras, R. Sarikaya, "Deep Belief Network based Semantic Taggers for Spoken Language Understanding", *In Proc. Interspeech*, Lyon, France, 2013.

[10] J.P. Robichaud, P. Crook, P. Xu, O. Z. Khan, R. Sarikaya, "Hypotheses Ranking for Robust Domain Classification and Tracking in Dialogue Systems," *In Proc. of Interspeech*, Singapore, September 2014.

[11] P. Crook, J.P. Robichaud, R. Sarikaya, "Multi-Language Hypotheses Ranking and Domain Tracking for Open Domain Dialogue Systems, *In Proc. Interspeech*, Dresden Germany, September 2015.

[12] P. Crook, et al., "Task Completion Platform: A self-serve multi-domain goal oriented dialogue platform", *Proc. NAACL*, San Diego, June 2016.

[13] O. Z. Khan, J.P. Robichaud, P. Crook, R. Sarikaya, Hypotheses Ranking and State Tracking for a Multi-Domain Dialog System using ASR Results, *Interspeech*, Dresden Germany, September, 2015.

[14] A. Bhargava, A. Celikyilmaz, D. H. Tur, and R. Sarikaya, "Easy Contextual Intent Prediction and Slot Detection," *ICASSP*, May 2013

[15] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network, in *Proc. of ICASSP*, 2014.

[16] C. Liu, P. Xu and R. Sarikaya, "Deep Contextual Language Understanding in Spoken Dialogue Systems, *In Proc. of Interspeech*, September, 2015.

[17] W. Ward and S. Issar, "Recent improvements in the CMU spoken language understanding system", *Proceedings of the workshop on Human Language Technology. Association for Computational Linguistics*, 1994.

[18] D. Hillard, A. Celikyilmaz, G. Tur, and D. Hakkani-Tür, "Learning Weighted Entity Lists from Web Click Logs for Spoken Language Understanding", *In Proc. of Interspeech* 2011.

[19] X. Liu, R. Sarikaya, "A Model Based Approach to Weight Dictionary Entities for Spoken Language Understanding, *IEEE SLT14*, Lake Tahoe, NV, 2014.

[20] F. Morbini, et.al., "A reranking approach for recognition and classification of speech input in conversational dialogue systems", *In Proc. of IEEE Workshop on Spoken Language Technology (SLT)*, December 2012, Miami, FL.

[21] M. Dinarelli, A. Moschitti, and G. Riccardi, "Discriminative reranking for spoken language understanding, *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 2, pp. 526539, February 2012.

[22] D. Boies, R. Sarikaya, A. Rochette, Z. Feizollahi, N. Ramesh,"Techniques for Updating Partial Dialog State", *United States Patent Application 20150095033*, 2013.

[23] A Celikyilmaz, Z Feizollahi, D Hakkani-Tr, R Sarikaya, "Resolving referring expressions in conversational dialogs for natural user interfaces", *In Proc. of EMNLP*, 2014.

[24] P. Xu, R. Sarikaya, "Exploiting Shared Information for Multi-Intent Natural Language Sentence Classification", *In Proc. of Interspeech*, Lyon, France, August, 2013.

[25] V. Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag, 1995.

[26] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", Proc. of ICML, 2001.

[27] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, K. Yu, "The Hidden Information State Model: a practical framework for POMDP-based spoken dialogue management.", *Computer Speech & Language*, 2009, 24 (2), pp.150-174.

[28] C. J. C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu, "Learning to Rank Using an Ensemble of Lambda-Gradient Models,, *Journal of Machine Learning Research*, vol. 14, pp. 25-35, 2011.

[29] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella, "PARADISE: A framework for evaluating spoken dialogue agents", *In Proc. of European chapter of the Association for Computational Linguistics*, 1997.

[30] J. Jiang, A. H. Awadallah, R. Jones, U. Ozertem, I. Zitouni, R. G. Kulkarni, and O. Z. Khan, "Automatic Online Evaluation of Intelligent Assistants", *Proc. of the 24th International World Wide Web Conference, ACM Association for Computing Machinery*, May 2015.

[31] K. Yao, J. Zweig, M. Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding", *Proc. of Interspeech*, 2013.